INTERNATIONALIZATION AND LOCALIZATION

WHAT IS INTERNATIONALIZATION AND LOCALIZATION?

Internationalization (i18n)

- Design software so it can be adapted to languajes without engineering changes

Localization(l10n)

- Adapt internationalized software to specific region or languaje

CREATION OF PROPERTIES FILES AND BUNDLE

I18nBundle.class

- Used to store and fetch strings in different languajes.
- Languaje code (ISO 639-1), country code (ISO 3166-1) and variant of locale

```
MyBundle.properties
MyBundle_de.properties
MyBundle_en_GB.properties
MyBundle_fr_CA_VAR1.properties
```

- How to use it

```
I18Bundle bundle = I18Bundle.createBundle(Gdx.files.
  internal("mybundles/bundle"))

TextButton play = newButton(bundle.get("play"));
```

- If a property file for the specified Locale does not exist createBundle will search for the closest match.

FETCHING LOCALIZED STRINGS

- Retrieve data from bundle with get(). `Textbutton play = bundle.get(“play”)`

- Retrieve data using Format
  - Translated strings containing parameters

```
String mission = myBundle.format("newMission", player.getName(),
nextLevel.getName());
```

- If no string for the given key can be found by the Get or Format method MissingResourceException will be thrown.

MESSAGE FORMAT

- Format string using java.text.MessageFormat Api.
  - Pattern can contain zero or more formats of the form {index, type, style} where the type and the style are optional

```
int planet = 7;
String event = "a disturbance in the Force";

String result =
MessageFormat.format("At {1,time} on {1,date}, there was {2} on
planet{0,number,integer}.", planet, new Date(), event);
```

- Out put is : At 12:30 PM on Jul 3, 2053, there was a disturbance in the Force on planet 7.

- Escape caracters in messageFormat is using single cuotes(‘) but in libgdx you just double it , for example {{0} is interpretted like {0}

- Formats are Localizable , typed data like dates, numbers and times will be expressed for the specific locale

PLURAL FORMS

- Plural forms are supported through the standard choice format provided by MessageFormat

```
collectedCoins=You collected {0,choice,0#no coins|1#one coin|
1<{0,number,integer} coins|100<hundreds of coins} along the path.

System.out.println(myBundle.format("collectedCoins", 0));
System.out.println(myBundle.format("collectedCoins", 1));
System.out.println(myBundle.format("collectedCoins", 32));
System.out.println(myBundle.format("collectedCoins", 117));

OUTPUT:

You collected no coins along the path.
You collected one coin along the path.
You collected 32 coins along the path.
You collected hundreds of coins along the path.
```

GWT LIMITATIONS AND COMPATIBILITY

- If using GWT- back end there are some limitations.
  - Simple format
    - Format's type and style are not supported and cannot be used or IllegalArgumentException will be thrown
    - Only {index} can be used.
  - Non localizable arguments
    - Formats are never localized , arguments passed to format are converted with toString() so it doesnt mind bundle´s locale

- If your application can run on both GWT and non-GWT back-ends, you should call I18NBundle.setSimpleFormat(true) when the application starts

This way all subsequent invocations of the factory method createBundle will create bundles having the same behavior on all back-ends.

MULTIPLE BUNDLES

- You can use multiple bundles , if your game has different levels , you can separate them in different bundles. Advantages are:
  - Code easier to read and maintain
  - Avoid huge bundles , less load in memory
  - Reduce memory by loading each bundle only when you need it